Today's lecture

Exact inference in graphical models.

- Variable Elimination
- Elimination as Graph Transformation
- Treewidth
- Sum-product belief propagation
- Max-product belief propagation
- Junction tree algorithm (during tutorial)

Figures from Koller book as well as Nowozin et al.

Inference: conditional probabilities

- Today we will look into exact inference in graphical models.
- We will focus on conditional probability queries

$$p(\mathbf{Y}|\mathbf{E}=\mathbf{e}) = \frac{P(\mathbf{Y},\mathbf{e})}{P(\mathbf{e})}$$

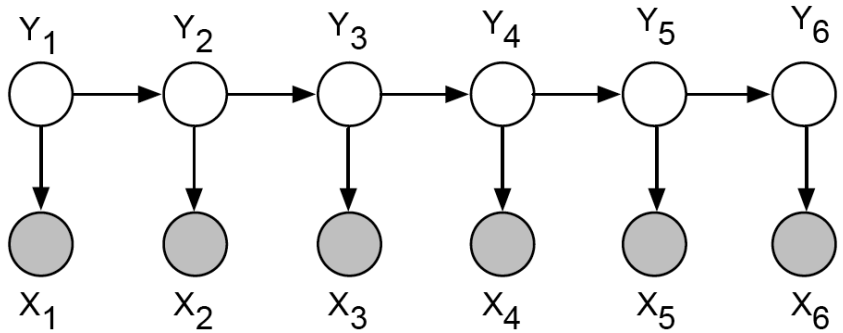
- Let $\mathbf{W} = \mathcal{X} - \mathbf{Y} - \mathbf{E}$ be the random variables that are neither the query nor the evidence. Each of the distributions can be computed by marginalizing over the other variables:

$$p(\mathbf{Y},\mathbf{e}) = \sum_{\mathbf{w}} P(\mathbf{Y},\mathbf{e},\mathbf{w}), \qquad P(\mathbf{e}) = \sum_{\mathbf{y},\mathbf{w}} P(\mathbf{y},\mathbf{e},\mathbf{w})$$

- Naively marginalizing over all unobserved variables requires an exponential number of computations. Why?
- Is there a more efficient algorithm?

Probabilistic inference in practice

- Probabilistic inference in NP-hard (both Directed and Undirected models)
- Should we give up?
- NP-hardness simply says that there exist difficult inference problems
- Real-world inference problems are not necessarily as hard as these worst-case instances
- It is easy to do inference in some graphs, e.g., inference in hidden Markov models (HMMs) and other tree-structured models



Variable Elimination (VE)

- Exact algorithm for probabilistic inference in **any** graphical model
- Running time will depend on the **graph structure**
- Uses **dynamic programming** to avoid enumerating all assignments
- Let's first look at computing marginal probabilities $p(X_i)$ in Directed models, and then generalize this to conditional queries on MRFs

Basic idea of variable elimination

- Let's start with a simple chain $A \rightarrow B \rightarrow C \rightarrow D$, and we want to compute $P(D)$
- This is nothing but computing a table
- By the chain rule we can write the joint distribution

$$p(A, B, C, D) = p(A)p(B|A)p(C|B)p(D|C)$$

- To compute $p(D)$, we can marginalize all other variables

$$p(D) = \sum_{a,b,c} p(A = a, B = b, C = c, D)$$

- This looks exponential, but $\cdots$

Let's be a bit more explicit...

$$
\begin{aligned}
& P(a^1)\ P(b^1 \mid a^1)\ P(c^1 \mid b^1)\ P(d^1 \mid c^1) \\
+\ & P(a^2)\ P(b^1 \mid a^2)\ P(c^1 \mid b^1)\ P(d^1 \mid c^1) \\
+\ & P(a^1)\ P(b^2 \mid a^1)\ P(c^1 \mid b^2)\ P(d^1 \mid c^1) \\
+\ & P(a^2)\ P(b^2 \mid a^2)\ P(c^1 \mid b^2)\ P(d^1 \mid c^1) \\
+\ & P(a^1)\ P(b^1 \mid a^1)\ P(c^2 \mid b^1)\ P(d^1 \mid c^2) \\
+\ & P(a^2)\ P(b^1 \mid a^2)\ P(c^2 \mid b^1)\ P(d^1 \mid c^2) \\
+\ & P(a^1)\ P(b^2 \mid a^1)\ P(c^2 \mid b^2)\ P(d^1 \mid c^2) \\
+\ & P(a^2)\ P(b^2 \mid a^2)\ P(c^2 \mid b^2)\ P(d^1 \mid c^2)
\end{aligned}
$$

$$
\begin{aligned}
& P(a^1)\ P(b^1 \mid a^1)\ P(c^1 \mid b^1)\ P(d^2 \mid c^1) \\
+\ & P(a^2)\ P(b^1 \mid a^2)\ P(c^1 \mid b^1)\ P(d^2 \mid c^1) \\
+\ & P(a^1)\ P(b^2 \mid a^1)\ P(c^1 \mid b^2)\ P(d^2 \mid c^1) \\
+\ & P(a^2)\ P(b^2 \mid a^2)\ P(c^1 \mid b^2)\ P(d^2 \mid c^1) \\
+\ & P(a^1)\ P(b^1 \mid a^1)\ P(c^2 \mid b^1)\ P(d^2 \mid c^2) \\
+\ & P(a^2)\ P(b^1 \mid a^2)\ P(c^2 \mid b^1)\ P(d^2 \mid c^2) \\
+\ & P(a^1)\ P(b^2 \mid a^1)\ P(c^2 \mid b^2)\ P(d^2 \mid c^2) \\
+\ & P(a^2)\ P(b^2 \mid a^2)\ P(c^2 \mid b^2)\ P(d^2 \mid c^2)
\end{aligned}
$$

- There is structure on the summation (repeated terms)
- Let's modify the computation to first compute

$$P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)$$

$$P(a^1)P(b^2|a^1) + P(a^2)P(b^2|a^2)$$

Let's be a bit more explicit...

- Let's modify the computation to first compute

$$P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)$$

and

$$P(a^1)P(b^2|a^1) + P(a^2)P(b^2|a^2)$$

- Then we get

$$\begin{array}{rlll} & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\ + & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{rlll} & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\ + & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\ + & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \end{array}$$

- We define $\tau_1 : Val(B) \rightarrow \Re, \quad \tau_1(b^i) = P(a^1)P(b^i|a^1) + P(a^2)P(b^i|a^2)$

Let's be a bit more explicit...

- We now have

$$\begin{array}{lll} & \tau_1(b^1) & P(c^1 \mid b^1) \quad P(d^1 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) \quad P(d^1 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) \quad P(d^1 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) \quad P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{lll} & \tau_1(b^1) & P(c^1 \mid b^1) \quad P(d^2 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) \quad P(d^2 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) \quad P(d^2 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) \quad P(d^2 \mid c^2) \end{array}$$

- We can once more reverse the order of the product and the sum and get

$$\begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^1 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{ll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) \quad P(d^2 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) \quad P(d^2 \mid c^2) \end{array}$$

- We have other repetitive patterns.

Let's be a bit more explicit...

- We define $\tau_2 : Val(C) \rightarrow \Re$, with

$$\begin{aligned} \tau_2(c^1) &= \tau_1(b^1)P(c^1|b^1) + \tau_1(b^2)P(c^1|b^2) \\ \tau_2(c^2) &= \tau_1(b^1)P(c^2|b^1) + \tau_1(b^2)P(c^2|b^2) \end{aligned}$$

- Thus we can compute the marginal $P(D)$ as

$$\begin{array}{rll} & \tau_2(c^1) & P(d^1 \mid c^1) \\ + & \tau_2(c^2) & P(d^1 \mid c^2) \\ \\ & \tau_2(c^1) & P(d^2 \mid c^1) \\ + & \tau_2(c^2) & P(d^2 \mid c^2) \end{array}$$

Even more explicit...

- The joint is

$$P(D) = \sum_{a,b,c} p(a,b,c,D) = \sum_{a,b,c} P(a)P(b|a)P(c|b)P(D|c)$$

$$\sum_c \sum_b \sum_a P(D|c)P(c|b)P(b|a)P(a)$$

- We can push the summation inside

$$P(D) = \sum_c P(D|c) \sum_b P(c|b) \underbrace{\sum_a \underbrace{P(b|a)P(a)}_{\psi_1(a,b)}}_{\tau_1(b)}$$

- Let's call $\psi_1(A,B) = P(A)P(B|A)$ and $\tau_1(B) = \sum_A \psi_1(A,B)$.
- We can define $\psi_2(B,C) = \tau_1(B)P(C|B)$ and $\tau_2(C) = \sum_B \psi_1(B,C)$.
- This procedure is dynamic programming: computation is inside out instead of outside in.

Complexity of a chain

- Generalizing to a chain $X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n$, and each node has k states.
- We can compute at each step $i = 1$ up to $n - 1$

$$P(X_{i+1}) = \sum_{x_i} P(X_{i+1}|x_i)P(x_i)$$

- We need to multiply $P(x_i)$ with each CPD $P(X_{i+1}|X_i)$ for each value of x_i.
- $P(X_i)$ has k values, and the CPD $P(X_{i+1}|X_i)$ has k^2 values.
- k^2 multiplications and $k(k-1)$ additions.
- The cost of the total chain is $\mathcal{O}(nk^2)$.
- In comparison, generating the full joint and summing it up has complexity $\mathcal{O}(k^n)$.
- We have done inference over the joint without generating it explicitly.

Summary so far

- Worst case analysis says that computing the joint is NP-hard.
- In practice due to the structure of the Bayesian network some subexpressions in the joint depend only on a subset of variables.
- We can catch up computations that are otherwise computed exponentially many times.
- This depends on our having a good **variable elimination ordering**

Sum-product Inference

- We want an algorithm to compute $P(\mathbf{Y})$ for directed and undirected models
- This can be reduced to the following **sum-product** inference task

$$\tau(\mathbf{Y}) = \sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi(\mathbf{z}_{Scope[\phi] \cap \mathbf{Z}}, \mathbf{y}_{Scope[\phi] \cap \mathbf{Y}}) \qquad \forall \mathbf{Y}$$

  where Φ is a set of potentials or factors

- For directed models, Φ is given by the conditional probability distributions for all variables,

$$\Phi = \{\phi_{X_i}\}_{i=1}^{n} = \{p(X_i | \mathbf{X}_{Pa(X_i)})\}_{i=1}^{n}$$

  where the sum is over the set $\mathbf{Z} = \mathcal{X} - \mathbf{Y}$

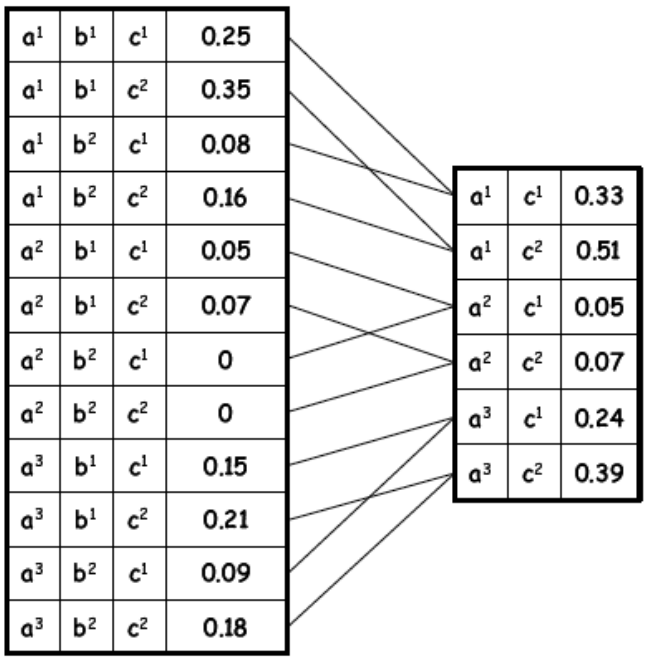
- For MRFs, the factors Φ correspond to the set of potentials
  - Sum product returns unnormalized distributions, so we normalize by dividing by $\sum_{\mathbf{y}} \tau(\mathbf{y})$

Variable elimination

- Let $\phi(\mathbf{X}, Y)$ be a factor where $\mathbf{X}$ is a set of variables and $Y \notin \mathbf{X}$
- **Factor marginalization** of ϕ over Y (also called "summing out Y in ϕ") to be a new factor

$$\tau(\mathbf{X}) = \sum_{Y} \phi(\mathbf{X}, Y)$$

- We only sum up the entries that $\mathbf{X}$ matches up



Sum-product Variable Elimination

- Order the variables $\mathbf{Z}$ (called the **elimination ordering**)
- Iteratively marginalize out variable Z_i one at a time
- For each i
  1. Multiply all factors that have Z_i in their scope, generating a new product factor
  2. Marginalize this product factor over Z_i, generating a smaller factor
  3. Remove the old factors from the set of all factors, and add the new one

Algorithm 9.1 Sum-Product Variable Elimination algorithm

Procedure Sum-Product-Variable-Elimination (
  Φ, // Set of factors
  $\boldsymbol{Z}$, // Set of variables to be eliminated
  $\prec$ // Ordering on $\boldsymbol{Z}$
)

Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
  $Z_i \prec Z_j$ iff $i < j$
for $i = 1, \ldots, k$
  $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
return ϕ^*

Procedure Sum-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)

$\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
$\Phi'' \leftarrow \Phi - \Phi'$
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
$\tau \leftarrow \sum_Z \psi$
return $\Phi'' \cup \{\tau\}$

Some properties

- If we sum out all the variables in a normalized distribution, what do we get?
- If we sum out all the variables in an unnormalized distribution, what do we get?
- Important property is that sum and product are commutative, and the product is associative, i.e., $(\phi_1\phi_2)\phi_3 = \phi_1(\phi_2\phi_3)$.
- Therefore, if $X \notin \mathit{Scope}(\phi_1)$ then

$$\sum_X(\phi_1\phi_2) = \phi_1 \sum_X \phi_2$$

Chain example again

- Let's look at the chain again

$$P(A, B, C, D) = \phi_A \phi_B \phi_C \phi_D$$

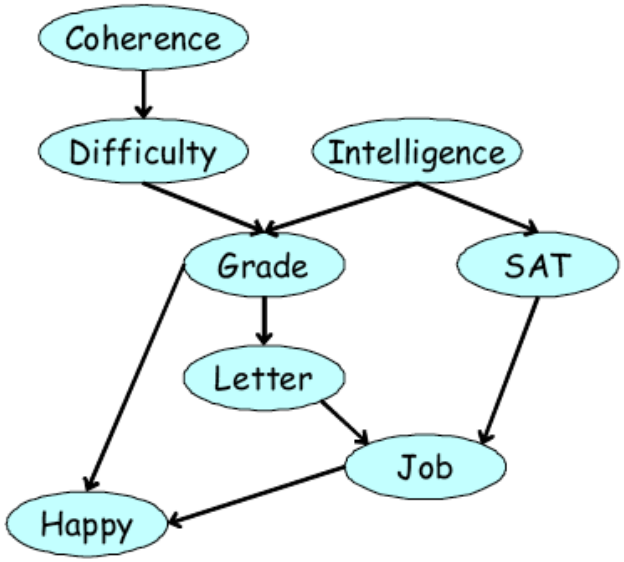
- The marginal distribution over D

$$\begin{aligned} P(D) &= \sum_{A,B,C} \phi_A \phi_B \phi_C \phi_D \\ &= \sum_C \left(\phi_D \sum_B \left(\phi_C \sum_A (\phi_B \phi_A) \right) \right) \end{aligned}$$

  where we have used the limited scope of the factors.

- Marginalizing involves taking the product of all CPDs and sum over all but the variables in the query.
- We can do this in any order we want; some more efficient than others.
- Effective as the scope is limited, we push in some of the summations.

Example of Directed Graph



- The joint distribution

$$p(C, D, I, G, S, L, H, J) = p(C)p(D|C)p(I)p(G|D, I)p(L|G)P(S|I)P(J|S, L)p(H|J, G)$$

with factors

$$\Phi = \{\phi_c(C), \phi_D(C, D), \phi_I(I), \phi_G(G, D, I), \phi_L(L, G), \phi_S(S, I), \phi_J(J, S, L), \phi_H(H, J, G)\}$$

- Let's do variable elimination with ordering $\{C, D, I, H, G, S, L\}$ on the board!

Example of Directed Graph

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \sum_D \psi_G(G,I,D) \underbrace{\sum_C \psi_C(C)\psi_D(D,C)}_{\tau_1(D)}$$

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \sum_I \psi_S(S,I)\psi_I(I) \underbrace{\sum_D \psi_G(G,I,D)\tau_1(D)}_{\tau_2(G,I)}$$

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \sum_H \psi_H(H,G,J) \underbrace{\sum_I \psi_S(S,I)\psi_I(I)\tau_2(G,I)}_{\tau_3(G,S)}$$

$$\sum_L \sum_S \psi_J(J,L,S) \sum_G \psi_L(L,G) \underbrace{\sum_H \psi_H(H,G,J)}_{\tau_4(G,J)} \tau_3(G,S)$$

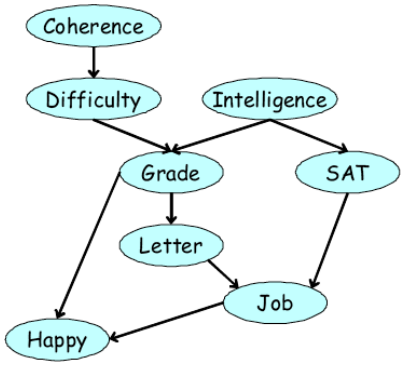
$$\sum_L \sum_S \psi_J(J,L,S) \underbrace{\sum_G \psi_L(L,G)\tau_4(G,J)\tau_3(G,S)}_{\tau_5(J,L,S)}$$

$$\sum_L \underbrace{\sum_S \psi_J(J,L,S)\tau_5(J,L,S)}_{\tau_6(J,L)}$$

$$\underbrace{\sum_L \tau_6(J,L)}_{\tau_7(J)}$$

Elimination Ordering

- We can pick any order we want, but some orderings introduce factors with much larger scope.



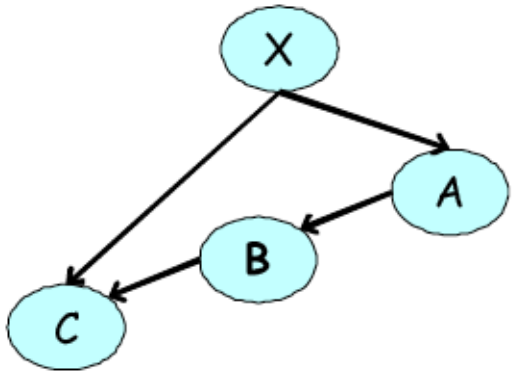
Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

- Alternative ordering...

Step	Variable eliminated	Factors used	Variables involved	New factor
1	G	$\phi_G(G,I,D)$, $\phi_L(L,G)$, $\phi_H(H,G,J)$	G,I,D,L,J,H	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_1(I,D,L,S,J,H)$	S,I,D,L,J,H	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S)$, $\tau_2(D,L,S,J,H)$	D,L,S,J,H	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	D,L,J,H	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	D,J,H	$\tau_5(D,J)$
6	C	$\tau_5(D,J)$, $\phi_D(D,C)$	D,J,C	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	D,J	$\tau_7(J)$

Semantics of Factors

- In the chain example the factors were marginal or conditional probabilities, but this is not true in general.



- The result of eliminating X is not a marginal or conditional probability of the network

$$\tau(A, B, C) = \sum_X P(X)P(A|X)P(C|B, X)$$

How to introduce evidence?

- But we wanted to answer *conditional* probability queries

$$p(\mathbf{Y}|\mathbf{E}=\mathbf{e}) = \frac{p(\mathbf{Y},\mathbf{e})}{p(\mathbf{e})}$$

- Apply variable elimination algorithm to the task of computing $P(\mathbf{Y},\mathbf{e})$.
- Replace each factor $\phi \in \Phi$ that has $\mathbf{E} \cap Scope[\phi] \neq 0$ with

$$\phi'(\mathbf{x}_{Scope[\phi]-\mathbf{E}}) = \phi(\mathbf{x}_{Scope[\phi]-\mathbf{E}}, \mathbf{e}_{\mathbf{E}\cap Scope[\phi]})$$

- Then eliminate the variables $\mathcal{X} - \mathbf{Y} - \mathbf{E}$. The return factor $\phi^*(\mathbf{Y})$ is $p(\mathbf{Y},\mathbf{e})$
- To obtain the conditional $p(\mathbf{Y}|\mathbf{e})$ normalize the resulting product of factors – the normalization constant is $p(\mathbf{e})$.

Sum-product VE for conditional distributions

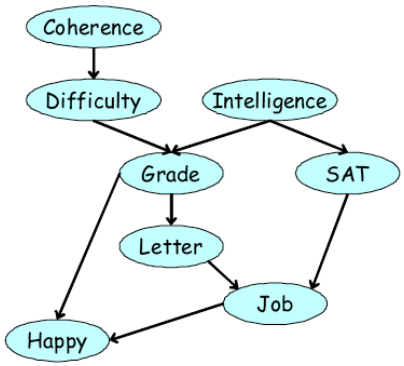
Algorithm 9.2 Using Sum-Product-Variable-Elimination for computing conditional probabilities.

Procedure Cond-Prob-VE (
  $\mathcal{K}$, // A network over $\mathcal{X}$
  $\boldsymbol{Y}$, // Set of query variables
  $\boldsymbol{E} = \boldsymbol{e}$ // Evidence
)

1 $\Phi \leftarrow$ Factors parameterizing $\mathcal{K}$
2 Replace each $\phi \in \Phi$ by $\phi[\boldsymbol{E} = \boldsymbol{e}]$
3 Select an elimination ordering $\prec$
4 $\boldsymbol{Z} \leftarrow = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$
5 $\phi^* \leftarrow$ Sum-Product-Variable-Elimination$(\Phi, \prec, \boldsymbol{Z})$
6 $\alpha \leftarrow \sum_{\boldsymbol{y} \in Val(\boldsymbol{Y})} \phi^*(\boldsymbol{y})$
7 **return** α, ϕ^*

Example

- We want to compute $P(J)$.

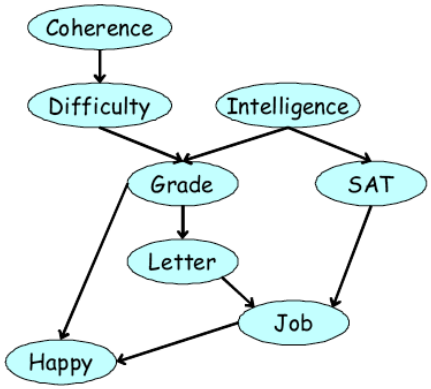


Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

- To compute $P(J|i^1,h^0)$

Step	Variable eliminated	Factors used	Variables involved	New factor
1'	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau'_1(D)$
2'	D	$\phi_G[I=i^1](G,D)$, $\phi_I[I=i^1]()$, $\tau'_1(D)$	G,D	$\tau'_2(G)$
5'	G	$\tau'_2(G)$, $\phi_L(L,G)$, $\phi_H[H=h^0](G,J)$	G,L,J	$\tau'_5(L,J)$
6'	S	$\phi_S[I=i^1](S)$, $\phi_J(J,L,S)$	J,L,S	$\tau'_6(J,L)$
7'	L	$\tau'_6(J,L)$, $\tau'_5(J,L)$	J,L	$\tau'_7(J)$

Complexity of Variable Elimination

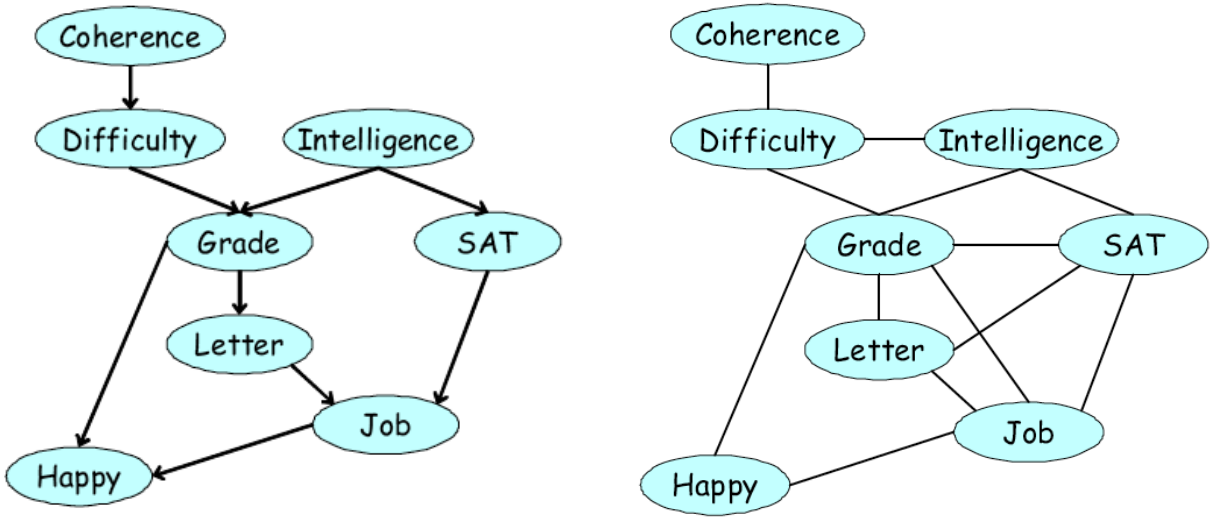


Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

- Let n be the number of random variables and m the number of initial factors.
- At each step we pick a variable X_i and multiply all factors involving the variable, resulting in a single factor ψ_i. The variable gets sum out of ψ_i, resulting in a new factor τ_i with $scope(\tau) = scope(\phi) - X_i$.
- Let N_i be the number of entries in the factor ψ_i, and let $N_{max} = \max_i N_i$.
- Therefore the total cost is $O(mk^{N_{max}})$, with $k = |Val(X)|$.
- Problem: N_{max} can be as large as n

Complexity in Graph terms

- Let's try to analyze the complexity in terms of the graph structure.
- The algorithm does not care if the graph is directed or undirected, only depends on the scope of the factors.
- Let $\mathcal{G}_\Phi$ be the undirected graph with one node per variable, where we have an edge iff there exists a factor $\phi \in \Phi$ such that $X_i, X_j \in Scope(\phi)$.
- Ignoring evidence, this is either the original MRF or the "moralized" directed graph



Elimination as Graph Transformation

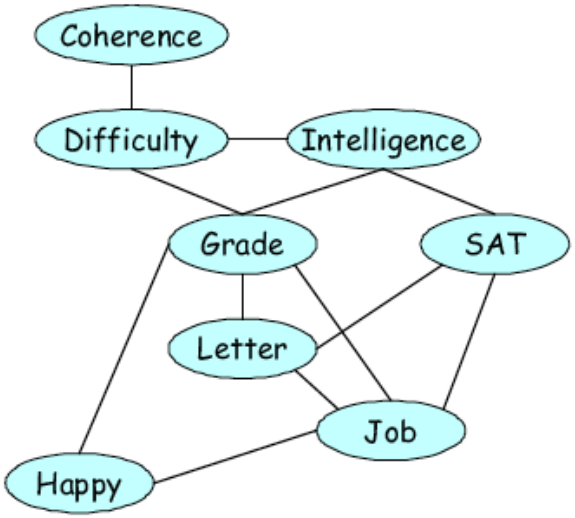
When a variable X is eliminated:

- We create a single factor ψ that contains X and all of the variables $\mathbf{Y}$ with which it appears in factors.
- We eliminate X from ψ, replacing it with a new factor τ that contains all of the variables $\mathbf{Y}$, but not X. Let's call this Φ_X.

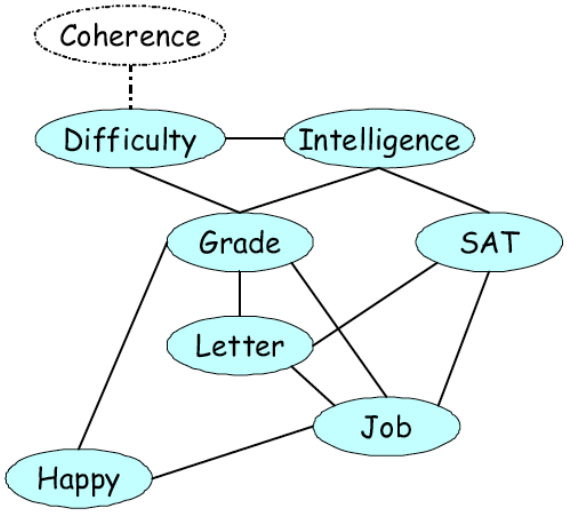
How does this modify the graph from $\mathcal{G}_\Phi$ to $\mathcal{G}_{\Phi_X}$?

- Constructing ψ generates edges between all of the variables $Y \in \mathbf{Y}$.
- Some of these edges were in $\mathcal{G}_\Phi$, some are new.
- The new edges are called **fill edges**.
- The step of removing X from ϕ to construct τ removes X and all it's incident edges from the graph.

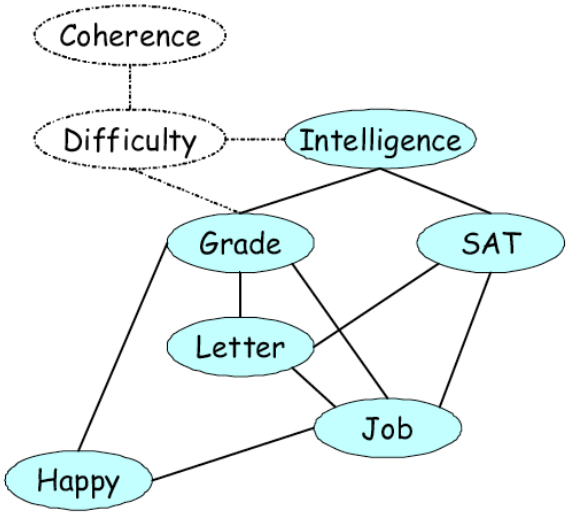
Example



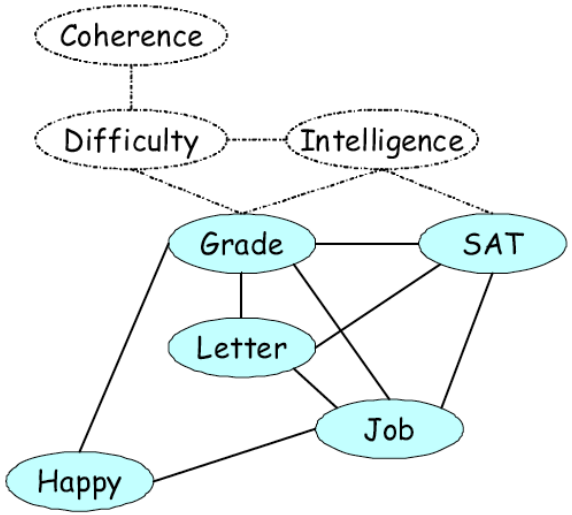
(Graph)



(Elim. C)



(Elim. D)

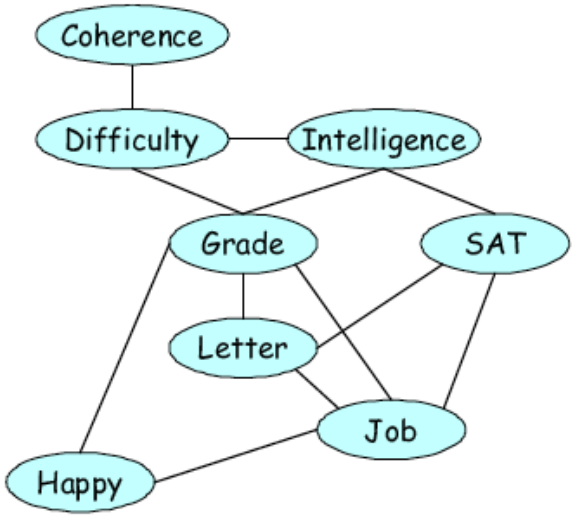


(Elim. I)

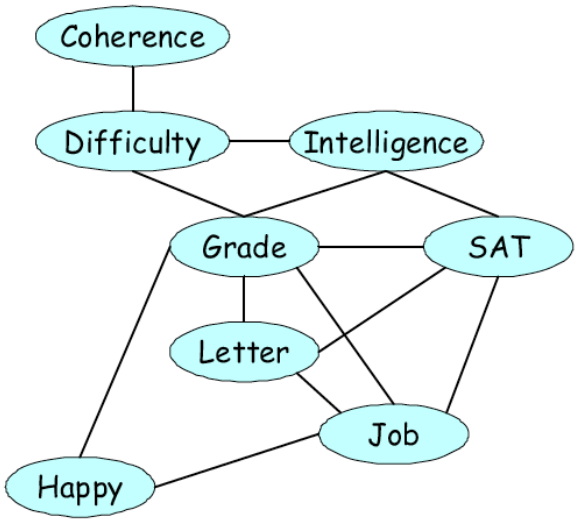
Induced Graph

- We can summarize the computation cost using a single graph that is the union of all the graphs resulting from each step of the elimination
- We call this the **induced graph** $\mathcal{I}_{\Phi,\prec}$ where $\prec$ is the elimination ordering
- N_{max} is the size of the largest clique in $\mathcal{I}_{\Phi,\prec}$
- The running time is $\mathcal{O}(mk^{N_{max}})$, exponential in the size of the largest clique of the induced graph

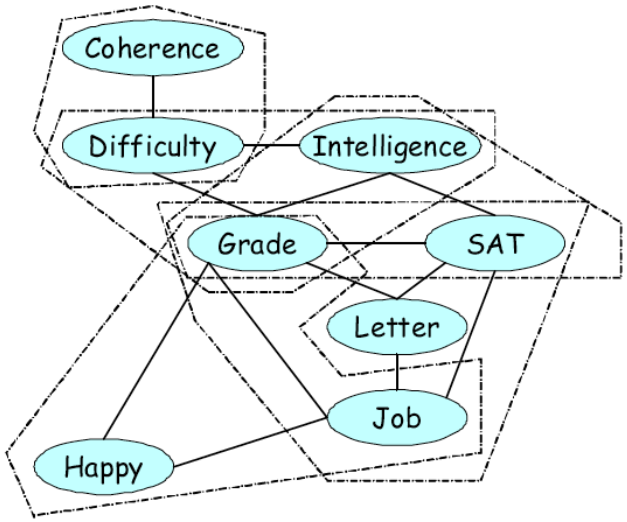
Example



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

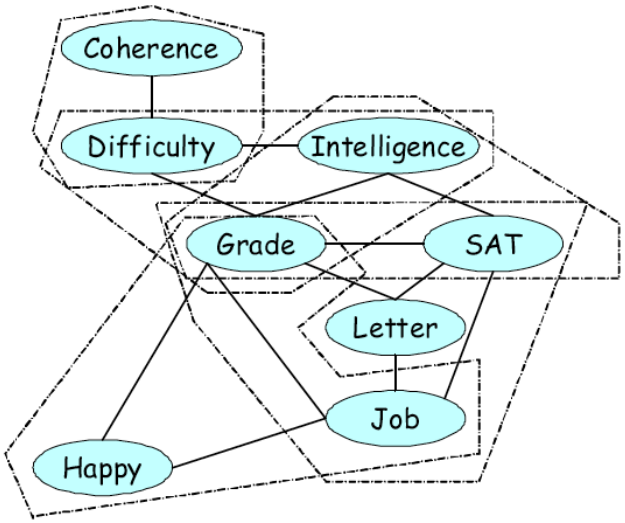


(Induced graph)



(Maximal Cliques)

Example



(Maximal Cliques)

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

(VE)

- The maximal cliques in $\mathcal{I}_{\mathcal{G},\prec}$ are

$$\begin{aligned} \mathbf{C}_1 &= \{C,D\} \\ \mathbf{C}_2 &= \{D,I,G\} \\ \mathbf{C}_3 &= \{G,L,S,J\} \\ \mathbf{C}_4 &= \{G,J,H\} \end{aligned}$$

Induced Width

- The **width** of an induced graph is #nodes in the largest clique -1.
- We define the **induced width** $w_{\mathcal{G},\prec}$ to be the width of the graph $\mathcal{I}_{\mathcal{G},\prec}$ induced by applying VE to $\mathcal{G}$ using ordering $\prec$.
- We define the **tree width** or **minimal induced width** of a graph $\mathcal{K}$ to be

$$w^*_{\mathcal{G}} = \min_{\prec} w(\mathcal{I}_{\mathcal{K},\prec})$$

- The treewidth provides a bound on the best running time achievable by VE on a distribution that factorizes over $\mathcal{G}$: $\mathcal{O}(mk^{w^*_{\mathcal{G}}+1})$
- Finding the best elimination ordering (also computing the tree width) is NP-hard
- Use heuristics to find a good elimination ordering

Choosing an Elimination Ordering

Set of possible heuristics:

- **Min-fill**: the cost of a vertex is the number of edges that need to be added to the graph due to its elimination.
- **Weighted-Min-Fill**: the cost of a vertex is the sum of weights of the edges that need to be added to the graph due to its elimination. Weight of an edge is the product of weights of its constituent vertices.
- **Min-neighbors**: The cost of a vertex is the number of neighbors it has in the current graph.
- **Min-weight**: the cost of a vertex is the product of weights (domain cardinality) of its neighbors.

Which one better?

- None of these criteria is better than others.
- Often try several

Sum-product belief propagation (BP)

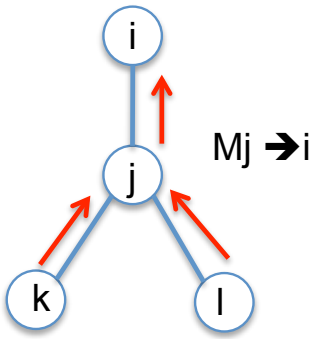
- What if you want to compute marginals for many variables, e.g., $p(X_i), \forall X_i \in \mathcal{X}$?
- We can run VE for each variable, but, can we do something more efficient?
- Consider a **tree**, we have

$$p(x_1, \cdots, x_n) = \frac{1}{Z} \prod_i \phi_i(x_i) \prod_{(i,j) \in \mathcal{T}} \phi_{i,j}(x_i, x_j)$$

- The **sum-product BP** computes all marginals with just two passes
- It is based on **message-passing** of "messages" (tables of partial summations) between neighboring vertices of the graph

Sum-product message

- The message sent from variable j to $i \in \mathcal{N}(j)$ is



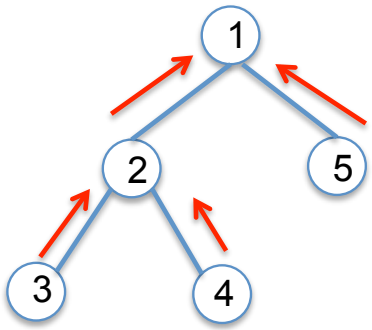
$$m_{j \to i}(x_i) = \sum_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{k \to j}(x_j)$$

- Each message $m_{j \to i}$ is a vector with one value for each state of x_i
- In order to compute $m_{j \to i}$, we must already have $m_{k \to j}(x_j)$ for $k \in \mathcal{N}(j) \setminus i$
- Thus we need a specific **ordering** of the messages

Example

- Suppose we want to compute $p(x_1)$, with x_1 the root

$$p(x_1) \propto \sum_{x_2,\cdots,x_n} \prod_i \phi_i(x_i) \prod_{(i,j)\in\mathcal{T}} \phi_{i,j}(x_i, x_j)$$



$$\begin{aligned} m_{5\to1}(x_1) &= \sum_{x_5} \phi_5(x_5)\phi_{15}(x_1, x_5) \\ m_{3\to2}(x_2) &= \sum_{x_3} \phi_3(x_3)\phi_{23}(x_2, x_3) \\ m_{4\to2}(x_2) &= \sum_{x_4} \phi_4(x_4)\phi_{24}(x_2, x_4) \end{aligned}$$

$$m_{2\to1}(x_1) = \sum_{x_2} \phi_2(x_2)\phi_{12}(x_1, x_2)m_{3\to2}(x_2)m_{4\to2}(x_2)$$

$$p(x_1) \propto \phi_1(x_1)m_{2\to1}(x_1)m_{5\to1}(x_1), \qquad Z = \sum_{x_1} p(x_1)$$

- Elimination algorithm in trees is equivalent to message passing
- What about we want $p(x_5)$? rerun the algorithm?

Belief Propagation (BP)

- Input: Tree $\mathcal{T}$ with potentials $\phi_i(x_i), \phi_{ij}(x_i, x_j) \forall (i,j) \in \mathcal{T}$
- Choose root r arbitrarily
- Pass messages from leafs to r
- Pass messages from r to leafs
- Compute

$$p(x_i) \propto \phi_i(x_i) \prod_{j \in \mathcal{N}(i)} m_{j \to i}(x_i), \qquad \forall i$$

- Running time is 2 times the cost of VE $= \mathcal{O}(nk^2)$ with $n = \#$nodes and $k = \#$states per node
- Numerically difficult: renormalized the messages to sum to 1 as constants are taking care by normalization constant later

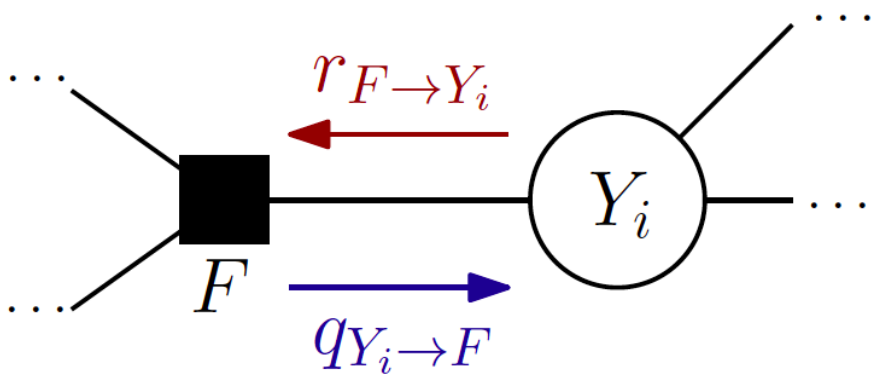
$$m^{new}_{j \to i}(x_i) = \frac{m^{old}_{j \to i}(x_i)}{\sum_{\hat{x}_i} m^{old}_{j \to i}(\hat{x}_i)}$$

- Even better to work in log domain

Generalization to Tree-Structured Factor Graphs

Iteratively updates and passes messages:

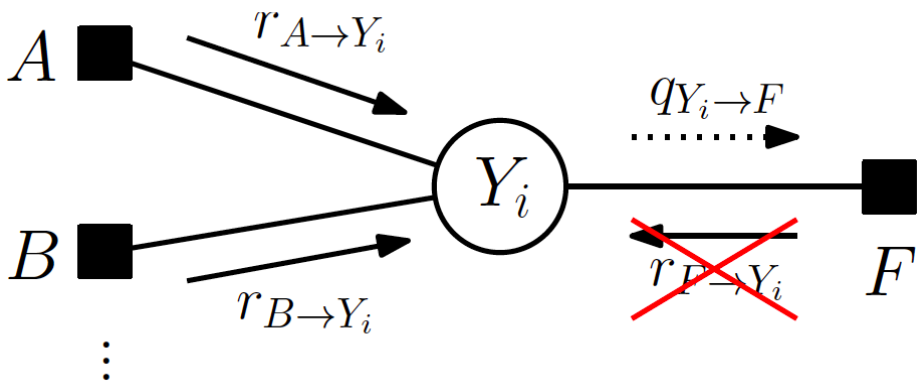
- $r_{F \to y_i} \in \Re^{\mathcal{Y}_i}$: factor to variable message
- $q_{y_i \to F} \in \Re^{\mathcal{Y}_i}$: variable to factor message



Variable to factor

- Let $M(i)$ be the factors adjacent to variable i, $M(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\}$
- Variable-to-factor message

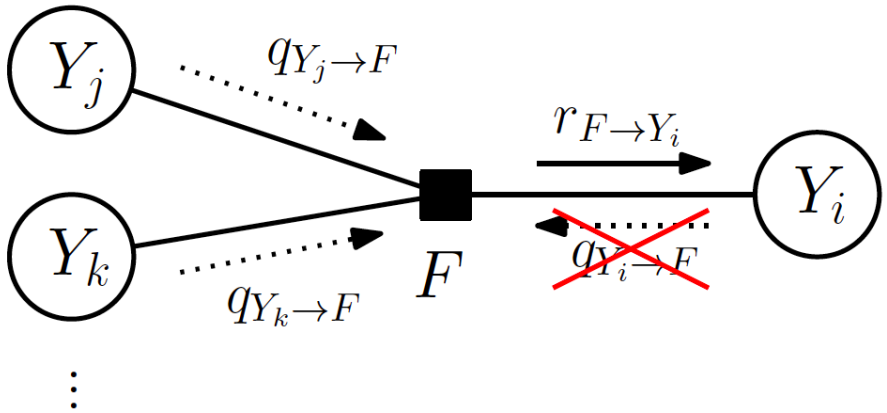
$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$



Factor to variable

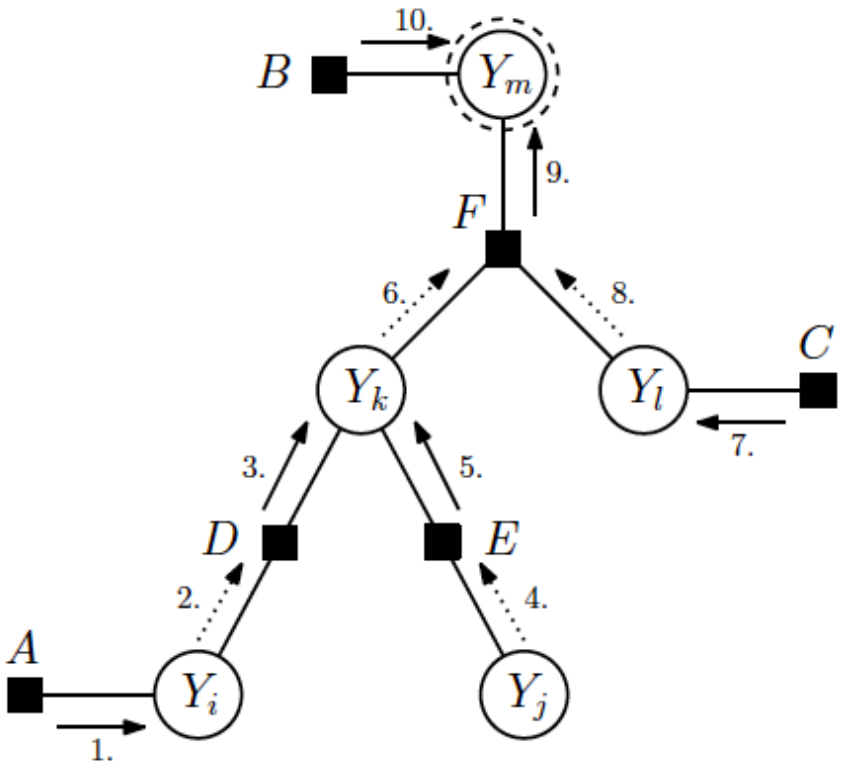
- Factor-to-variable message

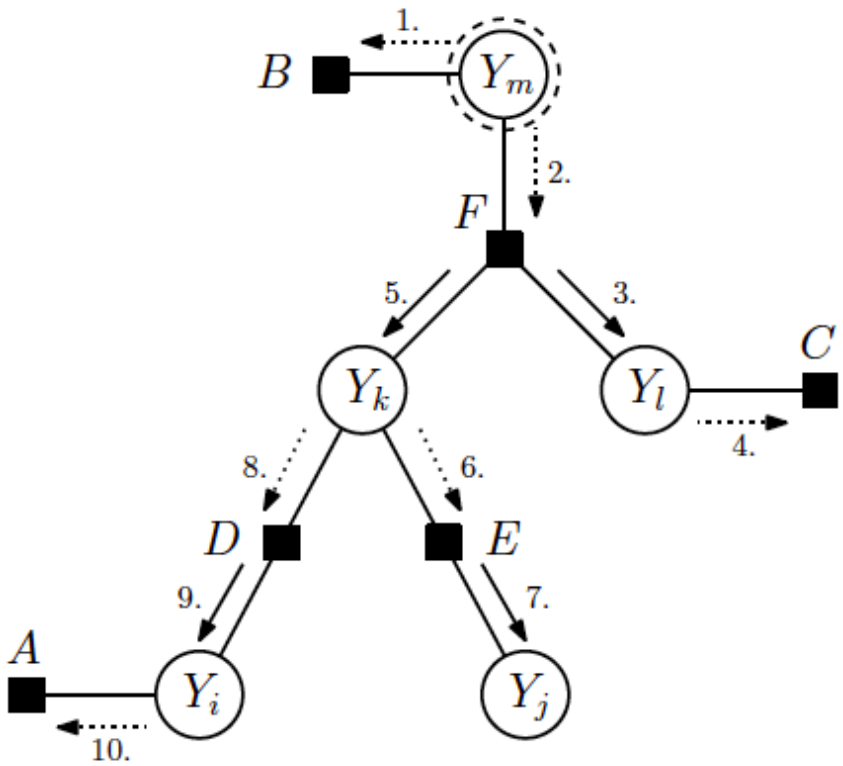
$$r_{F \to y_i}(y_i) = \log \sum_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \exp\left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y'_j \to F}(y'_j)\right)$$



Message Scheduling

1. Select one variable as tree root
2. Compute leaf-to-root messages
3. Compute root-to-leaf messages





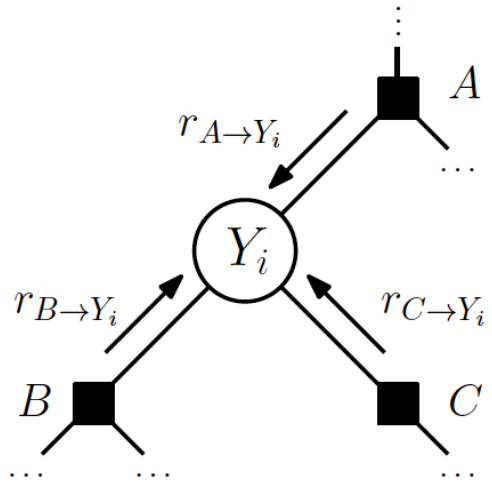
Computing marginals

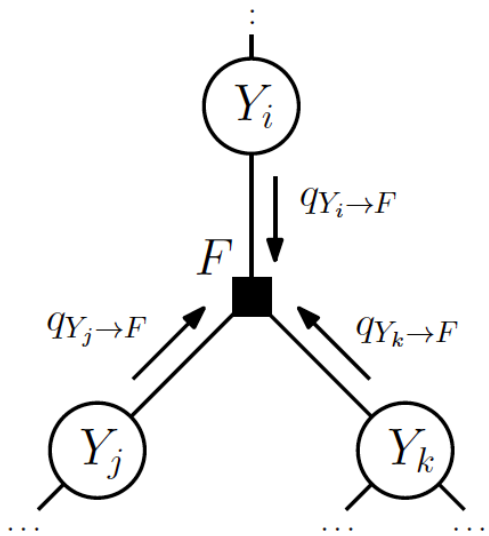
- Partition function can be evaluated at the root

$$\log Z = \log \sum_{y_r} \exp \left(\sum_{F \in M(r)} r_{F \to y_r}(y_r) \right)$$

- Marginal distributions, for each factor

$$\mu_F(y_F) = p(y_F) = \frac{1}{Z} \exp \left(\theta_F(y_F) + \sum_{i \in N(F)} q_{y_i \to F}(y_i) \right)$$





MAP Inference

- Recall the MAP inference task

$$\arg\max_{\mathbf{x}} p(\mathbf{x}), \qquad p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in C}\phi_c(\mathbf{x}_c)$$

  we assume any evidence has been subsumed into the potentials

- As the partition function is a constant we can alternatively

$$\arg\max_{\mathbf{x}} \prod_{c\in C}\phi_c(\mathbf{x}_c)$$

  This is the **max product** inference task

- Since the log is monotonic, let $\theta_c(\mathbf{x}_c) = \log\phi_c(\mathbf{x}_c)$

$$\arg\max_{\mathbf{x}} \sum_{c\in C}\theta_c(\mathbf{x}_c)$$

  This is called the **max-sum**

Semi-rings

- Compare the sum-product problem with the max-product (equivalently, max-sum in log space):

$$\text{sum-product} \qquad \sum_{\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{x}_c)$$

$$\text{max-product} \qquad \max_{\mathbf{x}} \sum_{c \in C} \theta_c(\mathbf{x}_c)$$

- Can exchange operators (+,*) for (max, +) and, because both are semirings satisfying associativity and commutativity, everything works!
- We get "max-product variable elimination" and "max-product belief propagation"

Simple Example

- Suppose we have a chain, $A - B - C - D$ and we want MAP

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- We can push the maximizations inside

$$\max_{a,b} \phi_{AB}(a,b) \max_{c} \phi_{BC}(b,c) \max_{d} \phi_{CD}(c,d)$$

or equivalently

$$\max_{a,b} \theta_{AB}(a,b) + \max_{c} \theta_{BC}(b,c) + \max_{d} \phi_{CD}(c,d)$$

- To find the actual maximizing assignment, we do a traceback

Max Product VE

Procedure Max-Product-VE (
  Φ, // Set of factors over $\boldsymbol{X}$
  $\prec$ // Ordering on $\boldsymbol{X}$
)

1 Let $X_1, \ldots, X_k$ be an ordering of $\boldsymbol{X}$ such that
2 $X_i \prec X_j$ if $i < j$
3 **for** $i = 1, \ldots, k$
4 $(\Phi, \phi_{X_i}) \leftarrow$ Max-Product-Eliminate-Var(Φ, X_i)
5 $x^* \leftarrow$ Traceback-MAP$(\{\phi_{X_i} : i = 1, \ldots, k\})$
6 **return** x^*, Φ // Φ contains the probability of the MAP

Procedure Max-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)

1 $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
2 $\Phi'' \leftarrow \Phi - \Phi'$
3 $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
4 $\tau \leftarrow \max_Z \psi$
5 **return** $(\Phi'' \cup \{\tau\}, \psi)$

Procedure Traceback-MAP (
  $\{\phi_{X_i} : i = 1, \ldots, k\}$
)

1 **for** $i = k, \ldots, 1$
2 $\boldsymbol{u}_i \leftarrow (x^*_{i+1}, \ldots, x^*_k)\langle Scope[\phi_{X_i}] - \{X_i\}\rangle$
3 // The maximizing assignment to the variables eliminated after X_i
4 $x^*_i \leftarrow \arg\max_{x_i} \phi_{X_i}(x_i, \boldsymbol{u}_i)$
5 // x^*_i is chosen so as to maximize the corresponding entry in the factor, relative to the previous choices $\boldsymbol{u}_i$
6 **return** x^*

Max-product belief propagation (for tree-structured MRFs)

- Same as sum-product BP except that the messages are now

$$m_{j\to i}(x_i) = \max_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in\mathcal{N}(j)\setminus i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node max

$$m_i(x_i) = \phi_i(x_i) \prod_{j\in\mathcal{N}(i)} m_{j\to i}(x_i), \qquad \forall i$$

- If the MAP assignment $\mathbf{x}^*$ is unique, we can find it by locally decoding each of the single node max-marginals

$$x_i^* = arg\max_{x_i} m_i(x_i)$$

Max-sum belief propagation (for tree-structured MRFs)

- Same as sum-product BP except that the messages are now

$$m_{j\to i}(x_i) = \max_{x_j} \theta_j(x_j) + \theta_{ij}(x_i, x_j) \sum_{k\in\mathcal{N}(j)\setminus i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node max

$$m_i(x_i) = \theta_i(x_i) + \sum_{j\in\mathcal{N}(i)} m_{j\to i}(x_i), \qquad \forall i$$

- If the MAP assignment $\mathbf{x}^*$ is unique, we can find it by locally decoding each of the single node max-marginals

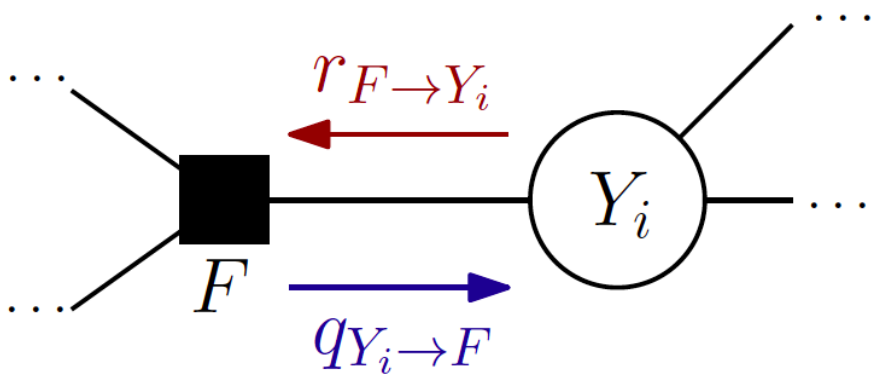
$$x_i^* = arg\max_{x_i} m_i(x_i)$$

- Working in log-space prevents numerical underflow/overflow

Factor Graph Max Product

Iteratively updates and passes messages:

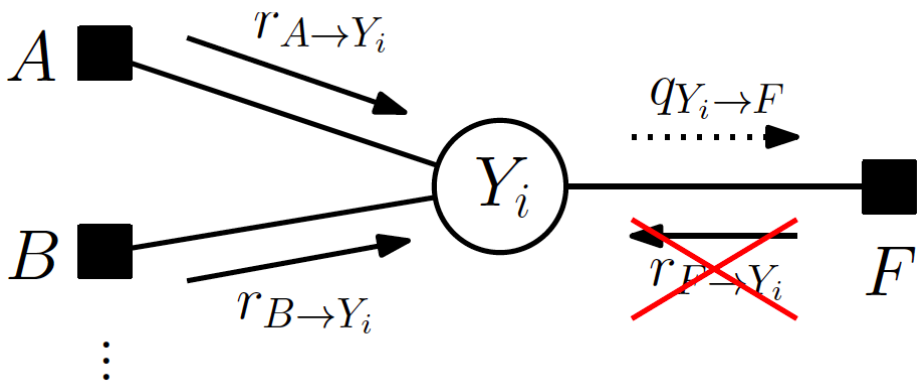
- $r_{F \to y_i} \in \Re^{\mathcal{Y}_i}$: factor to variable message
- $q_{y_i \to F} \in \Re^{\mathcal{Y}_i}$: variable to factor message



Variable to factor

- Let $M(i)$ be the factors adjacent to variable i, $M(i) = \{F \in \mathcal{F} : (i, F) \in \mathcal{E}\}$
- Variable-to-factor message

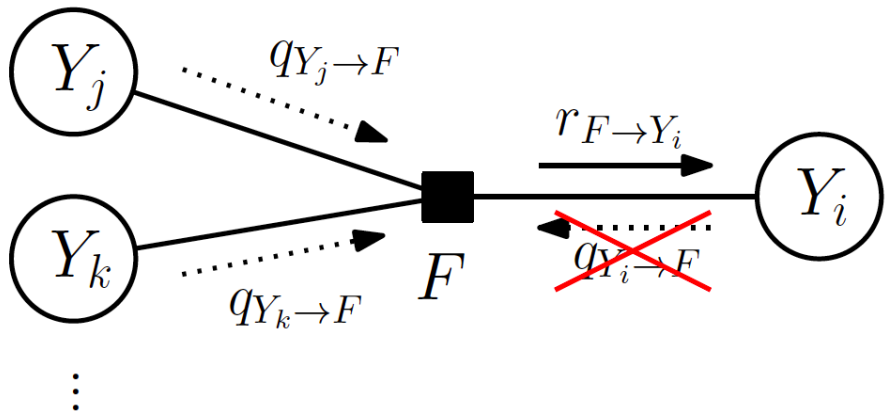
$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$



Factor to variable

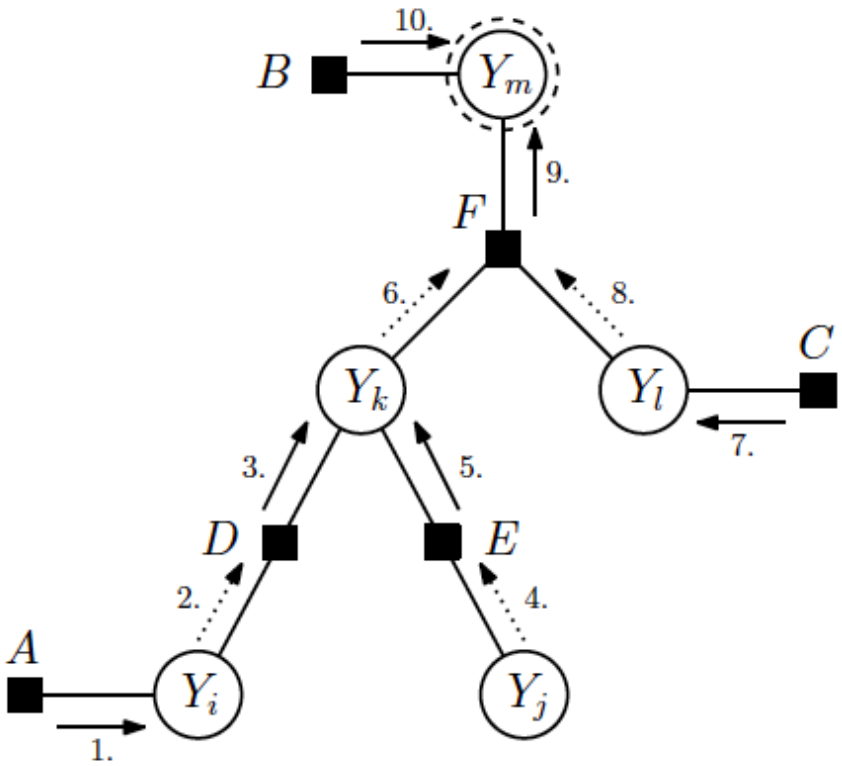
- Factor-to-variable message

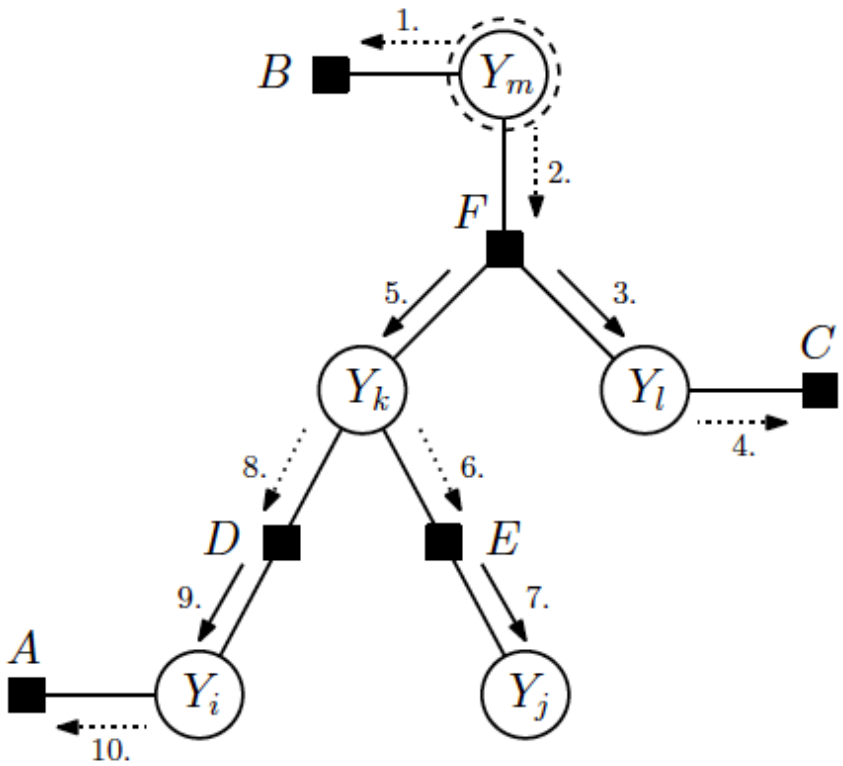
$$r_{F \to y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F}(y'_j) \right)$$



Message Scheduling

1. Select one variable as tree root
2. Compute leaf-to-root messages
3. Compute root-to-leaf messages





Max Product v Sum Product (log domain)

Max sum version of max-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \max_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y_j \to F}(y'_j) \right)$$

Sum-product

1. Compute leaf-to-root messages

$$q_{y_i \to F}(y_i) = \sum_{F' \in M(i) \setminus \{F\}} r_{F' \to y_i}(y_i)$$

2. Compute root-to-leaf messages

$$r_{F \to y_i}(y_i) = \log \sum_{y'_F \in \mathcal{Y}_F, y'_i = y_i} \exp \left(\theta(y'_F) + \sum_{j \in N(F) \setminus \{i\}} q_{y'_j \to F}(y'_j) \right)$$